UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/621,571 | 07/21/2000 | Rajesh Bordawekar | 13675(YOR9-2000-0365US1 | 4994 |

7590 02/24/2004

Richard L Catania
Scully Scott Murphy & Presser
400 Garden City Plaza
Garden City, NY 11530

| EXAMINER | |
|---|---|
| STEELMAN, MARY J | |
| **ART UNIT** | **PAPER NUMBER** |
| 2122 | 10 |

DATE MAILED: 02/24/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

| Office Action Summary | Application No | Applicant(s) | |
|---|---|---|---|
| | 09/621,571 | BORDAWEKAR ET AL. | |
| | Examiner | Art Unit | |
| | Mary J. Steelman | 2122 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *26 November 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *31-34* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *31-34* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *21 July 2000* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

## DETAILED ACTION

1. This Office Action is in response to RCE filed 11/26/2003. Claims 31-34 are amended. Claims 31-34 are pending.

### *Specification*

2. The use of the trademark JAVA / JAVA Virtual Machine has been noted in this application. It should be capitalized wherever it appears and be accompanied by the generic terminology.

Although the use of trademarks is permissible in patent applications, the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner which might adversely affect their validity as trademarks.

### *Claim Rejections - 35 USC § 112*

3. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

4. Claim33 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claim 33 is not clear, as it lacks a "step b)".

### *Claim Rejections - 35 USC § 103*

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 31, 33 and 34 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent 6,078,744 to Wolczko et al., in view of US Patent 6,367,012 to Atkinson et al.

Per claim 31:

Wolczko disclosed reuse of saved compilation data (static) that has been journaled. At col. 7, lines 18-22, Wolczko stated, "The record also distinguishes when the compilation unit has changed between the initial and subsequent compilations...many techniques exist for determining equivalence of the compilation unit." Col. 7, lines 43-53, "The 'compilation unit identification' field contains a value that identifies the compilation unit for the journal record...this value contains verification information used to verify that the compilation unit has not changed between an initial compilation and a subsequent compilation...verification information can be the source code of compilation unit, the checksum of the source code of the compilation unit of similar information." Col. 9, lines 57-60, "...the adaptive compiler is assured that the source program initially compiled is the same as the source program that is optimized." Col. 10, lines 41-66, "As each compilation unit is processed an 'equivalent unit in journal decision procedure determines whether the compilation unit in the source program...is equivalent to the compilation unit processed by the "initial phase compilation' process (by using the verification data stored in the 'information' field)...If the...decision procedure determines that the compilation units are not equivalent or that no information for the compilation unit has been journaled, the 'subsequent phase compilation' (dynamic) (therefore a mixed static and dynamic environment) process continues to 'generate intermediate compilation data' procedure...However, if the...decision procedure determines that the completion units are

equivalent...the 'subsequent phase compilation' process continues to 'read ICD data from journal procedure (static) ...instead of computing the ICD." (See Figure 6.) Wolczko disclosed "loading and executing the generated code" at col. 4, lines 41-45, "The JIT compiler compiles a method (a loaded method), a portion of a method, a statement of other source grouping just before the source groupings is first executed (load / execute the generated code). Subsequent calls to the source grouping re-execute the compiled target language for the host computer's ABI."

Wolczko failed to give specifics regarding how the equivalence is determined. However, Atkinson provided more detail on a certification or signature, incorporated in a program, file or code to assure authenticity and integrity, particularly for receiving over a network. (See Figure 3 and col. 6, lines 19-25) Col. 6, lines 30-33, "Code signing method assures the recipient of the identity of the source of file (i.e., its authenticity) (verifying that the intermediate or byte-code representation of the program is safe) and that the file was not modified after it was transmitted by that source (i.e., the integrity of file). Atkinson disclosed hashing (col. 6, lines 40-50), "...a cryptographic digest of "hash" of executable file is obtained (forming a secure hash describing the bye code) or computed (forming a secure hash describing the generated code). Standard hash functions are available...These functions take a...string and convert it to a fixed length output string...(called a cryptographic digest). This... "fingerprints" the file by producing a value that indicates whether a file submitted for download matches the original file (verifying that the secure hash of the byte-code matches the digitally signed secure hash for the byte-code / generated code). Hashing functions and the values they generate are secure..." See Figure 4 and col. 6, line 66 – col. 7, line 8, "...publisher digital certificate (digitally signed secure) and

publisher signature are attached or appended to or incorporated to executable file. Publisher signature and publisher digital certificate together form a keyed source confirmation with a secure representation..."

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Wolczko's invention that verifies that subsequent compilations have not changed, by specifying techniques such as those disclosed by Atkinson using secure hashing, digitally signing, and attaching a publisher signature and publisher digital certificate as methods to assure authenticity and integrity, because these are well known, secure ways to determine equivalency of files, an important feature for allowing code generated by the processes to be exchanged and trusted between machines.

Per claim 33:

Wolczko disclosed a mixed static and dynamic environment, col. 10, lines 41-66, "As each compilation unit is processed an 'equivalent unit in journal decision procedure determines whether the compilation unit in the source program...is equivalent to the compilation unit processed by the "initial phase compilation' (static) process (by using the verification data stored in the 'information' field)...If the...decision procedure determines that the compilation units are not equivalent or that no information for the compilation unit has been journaled, the 'subsequent phase compilation' (dynamic / updating statically generated code) (therefore a mixed static and dynamic environment) process continues to 'generate intermediate compilation data' (compiler generating the code for S (separately compiled code) to a) associate with method and data names, or signatures)" At col. 7, lines 18-22, Wolczko stated, "The record also distinguishes when the

compilation unit has changed between the initial and subsequent compilations (- c) check if byte codes associated with any names in the S (separately compiled code) relied upon by C (statically generated code) have changed by comparing)...many techniques exist for determining equivalence of the compilation unit." Col. 7, lines 43-53, "The 'compilation unit identification' field contains a value that identifies the compilation unit for the journal record...this value contains verification information used to verify that the compilation unit has not changed between an initial compilation and a subsequent compilation...verification information can be the source code of compilation unit, the checksum of the source code of the compilation unit of similar information." Col. 9, lines 57-60, "...the adaptive compiler is assured that the source program initially compiled is the same as the source program that is optimized." Col. 10, lines 41-66, "As each compilation unit is processed an 'equivalent unit in journal decision procedure determines whether the compilation unit in the source program...is equivalent to the compilation unit processed by the "initial phase compilation' process (by using the verification data stored in the 'information' field)...If the...decision procedure determines that the compilation units are not equivalent or that no information for the compilation unit has been journaled, the 'subsequent phase compilation' (- d)dynamically recompile the byte codes associated with C if any byte codes associated with S have changed) process continues to 'generate intermediate compilation data' procedure...However, if the...decision procedure determines that the completion units are equivalent...the 'subsequent phase compilation' process continues to 'read ICD data from journal procedure (static) ...instead of computing the ICD." (See Figure 6.) Wolczko disclosed "loading and executing the generated code" at col. 4, lines 41-45, "The JIT compiler compiles a method (a loaded method), a portion of a method, a statement of other source grouping just

before the source groupings is first executed (executing the generated code). Subsequent calls to the source grouping re-execute the compiled target language for the host computer's ABI."

Wolczko failed to give specifics regarding how the equivalence is determined. However, Atkinson provided more detail on a "secure hash of the name of the method and data names or signatures in S with the compiler for C recording the secure hash for the hash code". Atkinson disclosed hashing (col. 6, lines 40-50), "...a cryptographic digest of "hash" of executable file is obtained or computed (compiler associates with method and data names, or signatures, in S a secure hash, recording the secure hash). Standard hash functions are available...These functions take a...string and convert it to a fixed length output string...This... "fingerprints" the file by producing a value that indicates whether a file submitted for download matches the original file (comparing the secure hash of the names associated with S with the secure hash stored for this byte code in C). Hashing functions and the values they generate are secure..."

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Wolczko's invention that verifies that subsequent compilations have not changed, by specifying techniques such as those disclosed by Atkinson using secure hashing as a method to assure authenticity and integrity, because these are well known, techniques to enable trusting that files are safely exchanged between machines.

Per claim 34:

Wolczko disclosed a mixed static and dynamic environment, col. 10, lines 41-66, "As each compilation unit is processed an 'equivalent unit in journal decision procedure determines whether the compilation unit in the source program...is equivalent to the compilation unit

processed by the "initial phase compilation' (statically generated code) process (by using the verification data stored in the 'information' field)...If the...decision procedure determines that the compilation units are not equivalent or that no information for the compilation unit has been journaled, the 'subsequent phase compilation' (separately compiled ) (therefore a mixed static and dynamic environment) process continues to 'generate intermediate compilation data'. (use of statically generated code for some byte code that depends on some byte code that may be separately compiled)

Wolczko disclosed reuse of saved compilation data (statically generated code) that has been journaled. At col. 7, lines 18-22, Wolczko stated, "The record also distinguishes when the compilation unit has changed between the initial and subsequent compilations...many techniques exist for determining equivalence of the compilation unit." (check if any byte codes associated with any code S relied upon by C have changed) Col. 7, lines 43-53, "The 'compilation unit identification' field contains a value that identifies the compilation unit for the journal record...this value contains verification information used to verify that the compilation unit has not changed between an initial compilation and a subsequent compilation...verification information can be the source code of compilation unit, the checksum of the source code of the compilation unit of similar information." Col. 10, lines 41-66, "As each compilation unit is processed an 'equivalent unit in journal decision procedure determines whether the compilation unit in the source program...is equivalent to the compilation unit processed by the "initial phase compilation' process. Wolczko disclosed "interpreting the byte codes corresponding to C (statically generated code)" at col. 4, lines 41-45, "Subsequent calls to the source grouping re-

execute (interpret the byte codes corresponding to C) the compiled target language for the host computer's ABI."

Wolczko failed to give specifics regarding the use of a secure hash. However, Atkinson provided more detail on "associating with S a secure hash of the byte code associated with S" and "secure hash for the byte code corresponding to any S that affects the code generated for C". Atkinson disclosed hashing (col. 6, lines 40-50), "...a cryptographic digest of "hash" of executable file is obtained or computed (associating with S a secure hash of the byte code). Standard hash functions are available...These functions take a...string and convert it to a fixed length output string.... This... "fingerprints" the file by producing a value that indicates whether a file submitted for download matches the original file (secure hash for byte code corresponding to any S that affects the code generated for C). Hashing functions and the values they generate are secure..."

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Wolczko's invention that verifies that subsequent compilations have not changed, by specifying techniques such as those disclosed by Atkinson using secure hashing, because these are well known techniques to authenticate equivalency of files, an important feature for allowing code to be safely exchanged between machines.

7. Claim 32 is rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent 6,078,744 to Wolczko et al., in view of US Patent 6,317,872 to Gee et al.

Wolczko disclosed a mixed static and dynamic environment (col. 10, lines 41-66) that provides statically compiled code to a virtual machine, whereas the virtual machine would JIT

compile code dynamically if any change was detected. Wolczko did not disclose using the compiler for -maintaining symbolic entries for externally referenced symbols; -maintaining a mapping from locations in the generated code that reference external symbols to the symbolic entry for that symbol and the virtual machine, before the code is executed, performs the steps of -using the mapping and symbolic entries created by the compiler to generate direct references in the generated code to the externally referenced symbols that have been resolved by the virtual machine; -performing the default action on those external symbols that have not been resolved.

However, Gee disclosed an "improved method for resolving symbolic references in code generated by compiling source code" (Abstract, lines 5-7.) Gee disclosed (col. 22, lines 35 – col. 23, line 38, "When objects are created by the compiler, references are symbolic (by name) and are identified by entries in the aforementioned constant pool array (maintaining symbolic entries for externally referenced symbols)...Upon first access to the object, the instruction execution procedure will examine the field type and, upon determining that the field reference is unresolved, immediately perform the symbolic to logical addressing resolution function (performing the default action on those external symbols that have not been resolved)...to create resolved access information to be stored with the object(mapping from locations in the generated code that reference external symbols to the symbolic entry for that symbol)...Subsequent accesses to the object will examine the field_type value and determine that the references to the object have been resolved...The resolved access_flags field contains access control information that is associated with the object to implement security functions...In future invocations of the method, the index within the instruction stream will point to the method block pointer, which will then point to the method's method block (using the mapping and symbolic entries created by

the compiler to generate direct references). In this manner, a symbolic address only needs to be resolved upon the first invocation of a method.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Wolczko's invention used in a mixed static and dynamic environment to include features as disclosed by Gee, using mapping and symbolic entries to generate direct references because "load immediate" instructions are superior, quicker processes.

***Response to Arguments***

**(A) Applicant has argued, in substance, the following:**

As applicant has noted on page 6, 4th paragraph, of RCE filed 11/26/2003, "The prior art of record fails to disclose or teach the methods of claims 31-34 in a mixed static and dynamic environment.

**Examiner's Response:**

The Wolczko reference does provide a mixed static and dynamic environment. The invention uses previously compiled journaled code (static) as long as there has been no change. Otherwise, a JIT performs a compilation (dynamic). See figure 6 and col. 10, lines 41-66, "...an equivalent unit in journal decision procedure determines whether the compilation unit in the source program, currently being compiled, is equivalent...and whether any ICD (intermediate compilation data) is journaled for the compilation unit. If...procedure determines that the compilation units are not equivalent or that no information for the compilation unit has been journaled, the 'subsequent phase compilation' process continues to generate intermediate compilation data (dynamic)...However, if the equivalent unit in journal decision procedure determines that the completion units are equivalent and the journal contains an ICD for the

compilation unit, the subsequent phase compilation process continues to a 'read ICD data from journal' (static) procedure...instead of computing the ICD."

Furthermore, the phrase "in a mixed static and dynamic environment" also reads on JAVA code that is "compiled" statically into byte code, then "compiled" dynamically in a JIT virtual machine. This is well known in the art.

Additionally: In response to applicant's arguments, the recitation "in a mixed static and dynamic environment", in the claimed methods, has not been given patentable weight because the recitation only occurs in the preamble. A method preamble is generally not accorded any patentable weight where it merely recites the purpose of a process or the intended use of a structure, and where the body of the claim does not depend on the preamble for completeness but, instead, the process steps or structural limitations are able to stand alone. See *In re Hirao*, 535 F.2d 67, 190 USPQ 15 (CCPA 1976) and *Kropa v. Robie*, 187 F.2d 150, 152, 88 USPQ 478, 481 (CCPA 1951).

***Conclusion***

8. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (703) 305-4564. The examiner can normally be reached Monday through Thursday, from 7:00 A.M. to 5:30 P.M. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on (703) 305-4552.

The fax phone number is (703) 872-9306 for regular communications and for After Final communications. Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Mary Steelman

02/10/2004

TUAN DAM
SUPERVISORY PATENT EXAMINER